A Report

on

Creational Activity Learning

"Mess Management System"

Submitted By

Lande Sudam Narayan

Roll No: 94

PRN No : UCS22M1081

(TY Computer)

Guided By

Prof. P.M. Dhanrao

Department of Computer Engineering

SANJIVANI COLLEGE OF ENGINEERING

(An Autonomous Institution)

Kopargaon– 423 603, Maharashtra

In the academic year 2024-25

Sanjivani College of Engineering, Kopargaon

Department of Computer Engineering

CERTIFICATE

This is to certify that

Lande Sudam Narayan

(TY Computer)

Has successfully completed his CAL report on

" Mess Management System"

Towards the partial fulfillment of Bachelor's degree In Computer Engineering During the academic year 2024-25.

Prof. P.M. Dhanrao
[Subject Teacher)]

Dr. M.A.Jawale
[H.O.D. Comp Engg]

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to all those who helped me directly or indirectly in the successful completion of this project.

First and foremost, I would like to thank my project supervisor, **Prof. P.M. Dhanrao** for their invaluable guidance, timely suggestions, and continuous support throughout the development of the Mess Management System project. Their insights and encouragement inspired me to explore new dimensions of learning and enhanced the quality of my work significantly.

I would also like to extend my heartfelt thanks to the **Dr. M.A.Jawale** of Head of the Computer Department and all the faculty members of the Information Technology Department at Sanjivani College of Engineering for providing the necessary facilities and resources that enabled me to work efficiently on this project.

Special thanks to my family and friends for their constant motivation and emotional support throughout the course of this project. Their encouragement and belief in my capabilities helped me stay focused and confident.

Finally, I would like to thank all those who contributed directly or indirectly in making this project a success. This experience has enriched my knowledge and given me the confidence to take up future technical challenges with greater assurance.

Thank you all.

Mr. Lande Sudam
T.Y. Computer
Roll No: 94

INDEX

Abstract

The **Mess Management System** is a web-based application designed to streamline and automate the day-to-day operations of a mess or cafeteria. It aims to eliminate manual tasks such as menu planning, meal booking, billing, attendance tracking, and inventory control, making mess administration efficient and transparent.

The system provides different user roles, primarily Admin, Staff, and Students/Residents. Admins manage food menus, daily meals, user registrations, payments, and expenses. Students can register, log in, view menus, book/cancel meals, and check their monthly billing. Staff can update meal status and manage attendance. The system also supports automated monthly bill generation, attendance tracking, and inventory updates based on meal consumption.

This application enhances transparency, reduces food wastage, and promotes an eco-friendly digital environment by reducing paper-based records.

Key technologies used include:

- **Frontend:** HTML5, CSS3, JavaScript, JSP (JavaServer Pages)
- **Backend:** Java (JDK 8 or above), Servlets, JSP (JavaServer Pages)
- **Database:** MySQL, JDBC or Hibernate (for ORM)
- **Server:** Apache Tomcat

The project was developed using Eclipse IDE and deployed on Apache Tomcat Server. It involves structured database design and secure data handling practices to ensure the integrity and confidentiality of user information.

Introduction

In educational institutions, hostels, and corporate environments, mess or cafeteria management plays a crucial role in maintaining the daily nutrition and well-being of residents. However, traditional mess management systems are often manual, error-prone, and time-consuming, leading to issues like miscommunication, food wastage, billing errors, and poor transparency.

The Mess Management System is a web-based application developed to automate and simplify the management of mess operations. It provides an efficient digital platform for students or employees to book meals, view menus, and track their monthly expenses. Simultaneously, administrators can manage food schedules, monitor attendance, calculate bills, and keep track of inventory.

By replacing the conventional paper-based and manual tracking processes with a digital system, the application not only ensures better accuracy and accountability but also contributes to a sustainable and eco-friendly environment. It enhances user satisfaction through convenience, while reducing operational workload for the mess staff and management.

The system is built using Java technologies such as Servlets and JSP, integrated with MySQL for backend storage, and deployed on the Apache Tomcat server within the Eclipse IDE. It follows a structured MVC architecture to ensure separation of concerns, scalability, and maintainability.

Problem Statement

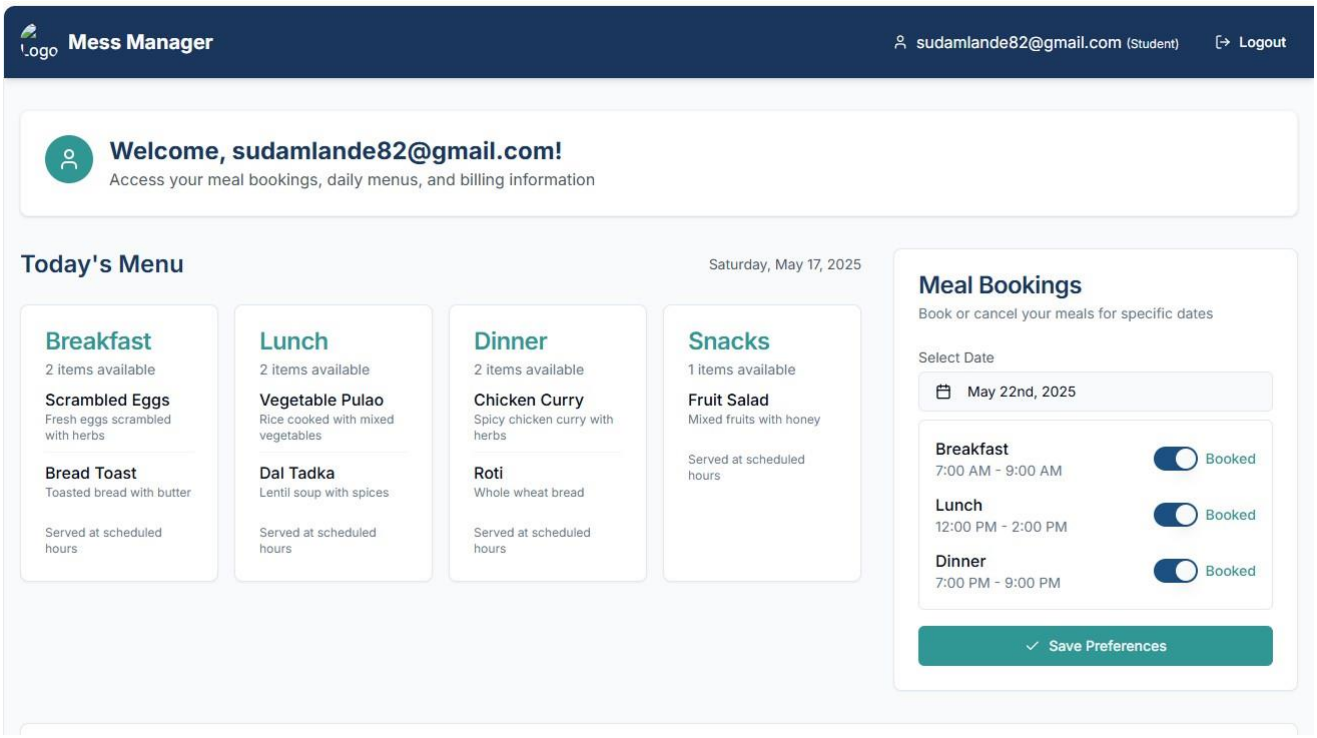
In many educational institutions, hostels, and corporate campuses, mess management is conducted manually, relying on registers or basic spreadsheets to track daily meals, attendance, billing, and inventory. This manual system often results in errors, inefficiencies, and mismanagement. Students or residents do not have a convenient way to view daily menus, book or cancel meals, or track their monthly consumption and expenses.

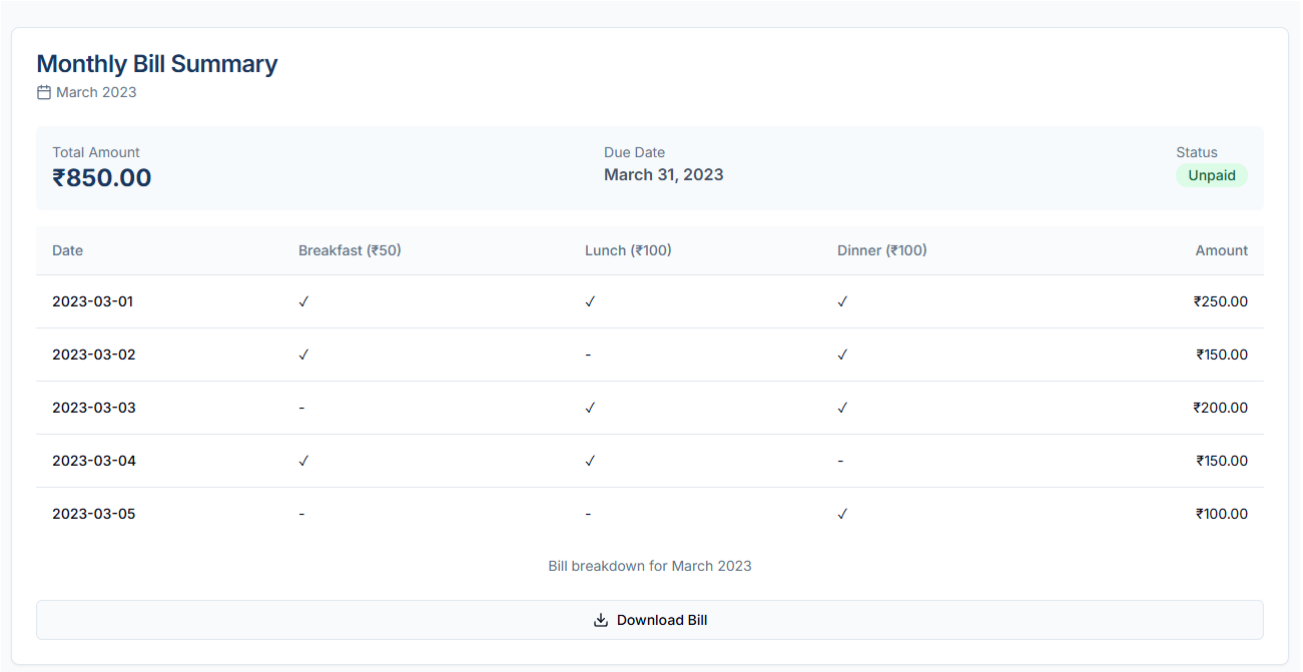
Administrators face challenges in maintaining accurate records, preventing food wastage, and generating timely reports for billing and inventory. The absence of a centralized and automated platform leads to communication gaps between mess users and management, delays in meal updates, and a lack of real-time monitoring.

To address these issues, there is a pressing need for a digital Mess Management System that can automate daily operations, provide user-friendly access for both residents and administrators, and ensure efficient, transparent, and scalable mess management.

Screenshots

Home Page ➔





Project Code

src/components/BillSummary.tsx

```
import React from 'react';
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from
'@/components/ui/card';
import { Button } from '@/components/ui/button';
import { Table, TableBody, TableCaption, TableCell, TableHead, TableHeader, TableRow }
from '@/components/ui/table';
import { Calendar, Download } from 'lucide-react';

interface BillItem {
  date: string;
  breakfast: boolean;
  lunch: boolean;
  dinner: boolean;
  totalAmount: number;
}

interface BillSummaryProps {
  month: string;
  year: number;
  billItems: BillItem[];
  totalAmount: number;
  dueDate: string;
}

const BillSummary: React.FC<BillSummaryProps> = ({
  month,
  year,
  billItems,
  totalAmount,
  dueDate
}) => {
  const mealRates = {
    breakfast: 50,
    lunch: 100,
    dinner: 100
  };

  const downloadBill = () => {
    // In a real application, this would generate a PDF or CSV
    console.log('Downloading bill...');
  };

  return (
    <Card className="animate-fade-in">
      <CardHeader>
```

```
        <CardTitle className="text-mess-blue">Monthly Bill Summary</CardTitle>
        <CardDescription className="flex items-center gap-1">
          <Calendar className="h-4 w-4" /> {month} {year}
        </CardDescription>
      </CardHeader>
      <CardContent  className="space-y-4">
        <div className="bg-mess-lightGray rounded-md p-4 flex flex-col sm:flex-row justify-
between gap-4">
          <div>
            <p className="text-sm text-muted-foreground">Total Amount</p>
            <p className="text-2xl font-bold text-mess-blue">₹{totalAmount.toFixed(2)}</p>
          </div>
          <div>
            <p className="text-sm text-muted-foreground">Due Date</p>
            <p className="text-md font-semibold text-mess-darkGray">{dueDate}</p>
          </div>
          <div>
            <p className="text-sm text-muted-foreground">Status</p>
            <span className="inline-flex items-center rounded-full bg-green-100 px-3 py-0.5
text-sm font-medium text-green-800">
              Unpaid
            </span>
          </div>
        </div>

        <div  className="overflow-x-auto">
          <Table>
            <TableCaption>Bill breakdown for {month} {year}</TableCaption>
            <TableHeader>
              <TableRow>
                <TableHead>Date</TableHead>
                <TableHead>Breakfast (₹{mealRates.breakfast})</TableHead>
                <TableHead>Lunch (₹{mealRates.lunch})</TableHead>
                <TableHead>Dinner (₹{mealRates.dinner})</TableHead>
                <TableHead className="text-right">Amount</TableHead>
              </TableRow>
            </TableHeader>
            <TableBody>
              {billItems.map((item) => (
                <TableRow key={item.date}>
                  <TableCell className="font-medium">{item.date}</TableCell>
                  <TableCell>{item.breakfast ? '✓' : '-'}</TableCell>
                  <TableCell>{item.lunch ? '✓' : '-'}</TableCell>
                  <TableCell>{item.dinner ? '✓' : '-'}</TableCell>
                  <TableCell className="text-right">₹{item.totalAmount.toFixed(2)}</TableCell>
                </TableRow>
              ))}
            </TableBody>
          </Table>
        </div>
```

```
      </CardContent>
      <CardFooter>
        <Button
          onClick={downloadBill}
          variant="outline"
          className="w-full flex items-center gap-2"
        >
          <Download className="h-4 w-4" />
          Download Bill
        </Button>
      </CardFooter>
    </Card>
  );
};

export default BillSummary;
```

src/components/BookingList.tsx

```
import React, { useState } from 'react';
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from
'@/components/ui/card';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Calendar } from '@/components/ui/calendar';
import { Popover, PopoverContent, PopoverTrigger } from '@/components/ui/popover';
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from
'@/components/ui/table';
import { Badge } from '@/components/ui/badge';
import { format } from 'date-fns';
import { Calendar as CalendarIcon, Check, X, Download } from 'lucide-react';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
'@/components/ui/select';

interface BookingData {
  id: string;
  studentId: string;
  studentName: string;
  date: Date;
  breakfast: boolean;
  lunch: boolean;
  dinner: boolean;
}

const BookingList: React.FC = () => {
  const [date, setDate] = useState<Date>(new Date());
  const [search, setSearch] = useState<string>('');
  const [mealFilter, setMealFilter] = useState<string>('all');

  // Mock data
```

```
const bookings: BookingData[] = [
  {
    id: '1',
    studentId: 'S12345',
    studentName: 'Rahul Sharma',
    date: new Date(),
    breakfast: true,
    lunch: true,
    dinner: false,
  },
  {
    id: '2',
    studentId: 'S12346',
    studentName: 'Priya Singh',
    date: new Date(),
    breakfast: false,
    lunch: true,
    dinner: true,
  },
  {
    id: '3',
    studentId: 'S12347',
    studentName: 'Amit Kumar',
    date: new Date(),
    breakfast: true,
    lunch: false,
    dinner: true,
  },
  {
    id: '4',
    studentId: 'S12348',
    studentName: 'Neha Gupta',
    date: new Date(),
    breakfast: true,
    lunch: true,
    dinner: true,
  },
];

// Filter bookings based on search and meal filter
const filteredBookings = bookings.filter(booking => {
const matchesSearch =
    booking.studentName.toLowerCase().includes(search.toLowerCase()) ||
    booking.studentId.toLowerCase().includes(search.toLowerCase());

  let matchesMeal = true;
  if (mealFilter === 'breakfast') matchesMeal = booking.breakfast;
  if (mealFilter === 'lunch') matchesMeal = booking.lunch;
  if (mealFilter === 'dinner') matchesMeal = booking.dinner;
```

```
  return matchesSearch && matchesMeal;
});

const downloadReport = () => {
  // In a real app, this would generate a CSV or PDF
  console.log('Downloading bookings report...');
};

// Count totals
const totalBreakfast = filteredBookings.filter(b => b.breakfast).length;
const totalLunch = filteredBookings.filter(b => b.lunch).length;
const totalDinner = filteredBookings.filter(b => b.dinner).length;

return (
  <div className="space-y-6 animate-fade-in">
    <Card>
      <CardHeader>
        <CardTitle className="text-mess-blue">Meal Bookings</CardTitle>
        <CardDescription>
          View and manage student meal bookings
        </CardDescription>
      </CardHeader>
      <CardContent  className="space-y-4">
        <div className="flex flex-col sm:flex-row gap-4">
          <div  className="flex-1">
            <Popover>
              <PopoverTrigger asChild>
                <Button
                  variant="outline"
                  className="w-full justify-start text-left font-normal"
                >
                  <CalendarIcon className="mr-2 h-4 w-4" />
                  {format(date, 'PPP')}
                </Button>
              </PopoverTrigger>
              <PopoverContent className="w-auto p-0">
                <Calendar
                  mode="single"
                  selected={date}
                  onSelect={(newDate) => newDate && setDate(newDate)}
                  initialFocus
                />
              </PopoverContent>
            </Popover>
          </div>

          <div  className="flex-1">
            <Input
              placeholder="Search by name or ID"
              value={search}
```

```
      onChange={(e) => setSearch(e.target.value)}
      className="w-full"
    />
  </div>

  <div className="w-40">
    <Select value={mealFilter} onValueChange={setMealFilter}>
      <SelectTrigger>
        <SelectValue placeholder="Filter by meal" />
      </SelectTrigger>
      <SelectContent>
        <SelectItem value="all">All Meals</SelectItem>
        <SelectItem value="breakfast">Breakfast</SelectItem>
        <SelectItem value="lunch">Lunch</SelectItem>
        <SelectItem value="dinner">Dinner</SelectItem>
      </SelectContent>
    </Select>
  </div>

  <Button
    variant="outline"
    className="flex items-center gap-2"
    onClick={downloadReport}
  >
    <Download className="h-4 w-4" />
    Export
  </Button>
</div>

<div className="flex flex-wrap gap-3 pb-2">
  <Badge variant="outline" className="bg-mess-lightGray">
    Breakfast: {totalBreakfast}
  </Badge>
  <Badge variant="outline" className="bg-mess-lightGray">
    Lunch: {totalLunch}
  </Badge>
  <Badge variant="outline" className="bg-mess-lightGray">
    Dinner: {totalDinner}
  </Badge>
  <Badge variant="outline" className="bg-mess-lightGray">
    Total Students: {filteredBookings.length}
  </Badge>
</div>

<div  className="overflow-x-auto">
  <Table>
    <TableHeader>
      <TableRow>
        <TableHead>Student ID</TableHead>
        <TableHead>Name</TableHead>
```

```
              <TableHead className="text-center">Breakfast</TableHead>
              <TableHead className="text-center">Lunch</TableHead>
              <TableHead className="text-center">Dinner</TableHead>
            </TableRow>
          </TableHeader>
          <TableBody>
            {filteredBookings.length === 0 ? (
              <TableRow>
                <TableCell colSpan={5} className="text-center text-muted-foreground">
                  No bookings found
                </TableCell>
              </TableRow>
            ) : (
              filteredBookings.map((booking) => (
                <TableRow key={booking.id}>
                  <TableCell>{booking.studentId}</TableCell>
                  <TableCell>{booking.studentName}</TableCell>
                  <TableCell className="text-center">
                    {booking.breakfast ? (
                      <Check className="h-4 w-4 text-green-600 mx-auto" />
                    ) : (
                      <X className="h-4 w-4 text-red-600 mx-auto" />
                    )}
                  </TableCell>
                  <TableCell className="text-center">
                    {booking.lunch ? (
                      <Check className="h-4 w-4 text-green-600 mx-auto" />
                    ) : (
                      <X className="h-4 w-4 text-red-600 mx-auto" />
                    )}
                  </TableCell>
                  <TableCell className="text-center">
                    {booking.dinner ? (
                      <Check className="h-4 w-4 text-green-600 mx-auto" />
                    ) : (
                      <X className="h-4 w-4 text-red-600 mx-auto" />
                    )}
                  </TableCell>
                </TableRow>
              ))
            )}
          </TableBody>
        </Table>
      </div>
    </CardContent>
  </Card>
 </div>
);
};
```

```
export default BookingList;
```

Conclusion

The **Mess Management System** successfully automates the essential operations of a mess facility, replacing traditional manual processes with a streamlined digital platform. By integrating meal booking, attendance tracking, menu management, billing, and inventory control into one accessible web application, it significantly improves accuracy, transparency, and efficiency.

This system not only reduces the administrative workload but also enhances the user experience for residents by providing convenient access to important mess-related services. The adoption of this system helps minimize food wastage, optimizes resource management, and fosters a sustainable environment.

Overall, the Mess Management System demonstrates how technology can effectively transform routine mess operations into an organized, error-free, and user-friendly process, benefiting both the administration and the consumers alike.

References

1. **Core Java Volume I – Fundamentals**, by Cay S. Horstmann and Gary Cornell, 11th Edition, Pearson Education, 2018.

2. **Head First Servlets and JSP**, by Bryan Basham, Kathy Sierra, Bert Bates, 2nd Edition, O'Reilly Media, 2008.

3. **Java Servlet & JSP Cookbook**, by Bruce W. Perry, O'Reilly Media, 2004.

4. Oracle Official Documentation for **Java Servlet Technology**: https://javaee.github.io/javaee-spec/javadocs/javax/servlet/package-summary.html

5. Oracle Official Documentation for **JSP (JavaServer Pages)**: https://javaee.github.io/javaee-spec/javadocs/javax/servlet/jsp/package-summary.html

6. **MySQL 8.0 Reference Manual** — https://dev.mysql.com/doc/refman/8.0/en/

7. TutorialsPoint, **Java Servlet Tutorial** — https://www.tutorialspoint.com/servlets/index.htm

8. TutorialsPoint, **JSP Tutorial** — https://www.tutorialspoint.com/jsp/index.htm

9. GeeksforGeeks, **Mess Management System Project** (for conceptual understanding): https://www.geeksforgeeks.org/mess-management-system-java/

10. **MVC Architecture Pattern** — https://www.geeksforgeeks.org/mvc-design-pattern/